```
1
2
3   from a0_items import *
4
5
6   '''
7   # initializing list
8   test_list = [1, 5, 6, 7, 4, 2323]
9
10  # printing original list
11  print("The original list is : " + str(test_list))
12
13  # using list indexing
14  # to get last element of list
15  res = [test_list[-1]]
16
17  # printing result
18  print("The last element of list are : " + str(res))
19  '''
20
21
22  ################################################################################################################################
23  #### Multi-processing
24  ################################################################################################################################
25
26  import multiprocessing
27  from multiprocessing import Process
28
29  print("Number of cpu : ", multiprocessing.cpu_count())
30
31
32
33  def fun_0():
34
35      for x in range(100):
36         print ("Shane")
37
38
39  def fun_1():
40
41     for x in range(500):
42        print ("Lucy")
43
44
45  def fun_2():
46
47     for x in range(500):
48        print ("ALEX")
49
50
51  def test123():
52
53      p0 = Process(target=fun_0)
54      p0.start()
55
56      p1 = Process(target=fun_1)
57      p1.start()
58
59      p2 = Process(target=fun_2)
60      p2.start()
61
62
63  '''
64  #must run multi-proce using if __name__ == '__main__'
65  if __name__ == '__main__':
```

```
    test123()
'''


################################################################################################################################
#### Concurrent futures
################################################################################################################################


from concurrent.futures import ThreadPoolExecutor, wait
import concurrent.futures
import time



Num1 = [1, 3, 4, 8]
Num2 = [8, 1, 3, 7]
Words1 = ['hey', 'yo', 'blue', 'no']
Words2 = ['Ho' , 'Ming', 'Peter', 'John']



def print_sth (var1, var2, var3, var4):

   print (var1 + var2, var3, var4)



# https://www.youtube.com/watch?v=IEEhzQoKtQU
# https://www.packetswitch.co.uk/what-is-concurrent-futures-and-how-can-it-boost-your-python-performance/
# https://superfastpython.com/processpoolexecutor-common-errors/
def concur_0_to_3 ():

   with concurrent.futures.ProcessPoolExecutor() as executor:

      executor.map(print_sth, Num1[2:4], Num2[2:4], Words1[2:4], Words2[2:4])

   t2 = time.perf_counter()
   print (f' finished in {t2} seconds')


#concur_0_to_3 ()




################################################################################################################################
# Concurrent inside concurrent
################################################################################################################################

LIST_a = ['a1', 'a2', 'a3','a4', 'a5']
LIST_b = ['b1', 'b2', 'b3','b4', 'b5']
LIST_c = ['c1', 'c2', 'c3','c4', 'c5']

def func_a (var_a):

   print (var_a)


def func_b (var_b):

```

```
127     print (var_b)
128
129
130 def func_c (var_c):
131
132     print (var_c)
133
134
135
136 def CF_a ():
137
138     with concurrent.futures.ProcessPoolExecutor() as executor:
139
140           executor.map(func_a, LIST_a)
141
142     t2 = time.perf_counter()
143     print (f' finished in {t2} seconds')
144
145
146
147 def CF_b ():
148
149     with concurrent.futures.ProcessPoolExecutor() as executor:
150
151           executor.map(func_b, LIST_b)
152
153     t2 = time.perf_counter()
154     print (f' finished in {t2} seconds')
155
156
157
158 def CF_c ():
159
160     with concurrent.futures.ProcessPoolExecutor() as executor:
161
162           executor.map(func_c, LIST_c)
163
164     t2 = time.perf_counter()
165     print (f' finished in {t2} seconds')
166
167
168
169 def CF_sequence ():
170
171     CF_a ()
172     CF_b ()
173     CF_c ()
174
175
176
177 def CF_ALL ():
178
179     with ThreadPoolExecutor(3) as ex:
180        futures = []
181        futures.append(ex.submit(CF_a))
182        futures.append(ex.submit(CF_b))
183        futures.append(ex.submit(CF_c))
184
185
186
187 #must run concurrency using if __name__ == '__main__'
188 if __name__ == '__main__':
189
190     CF_a()
```